AMENDMENTS TO THE CLAIMS

This listing of claims will replace all prior versions and listings of claims in the application:

LISTING OF CLAIMS:

1. (currently amended): A method of verification for a design, comprising:

providing a description of said design;

specifying correctness criteria for said design, wherein said correctness criteria are expressed as one or more correctness properties;

abstracting said design description to provide an abstract model of said design;

generating a witness graph for said one or more correctness properties based on a deterministic analysis of said abstract model where the deterministic analysis serves to modify the abstract model by over-approximating states and transitions which capture all witnesses or counterexamples demonstrating said one or more correctness properties;

~~determining a conclusive result from the set consisting of property violation and property satisfaction, when said witness graph is empty;~~ and

where the deterministic analysis does not produce a conclusive result, generating a testbench automatically ~~when said witness graph is non empty~~ from said witness graph for~~, and~~ performing simulation with said testbench, where the testbench generates simulation test vectors for the simulation that target the over-approximated states and transitions in the witness graph when searching for a concrete witness or counterexample with respect to said correctness criteria~~;~~

~~wherein, when a property refers to universal path quantification, said witness graph includes paths demonstrating only said property violation, defining counter-examples;~~

~~wherein, when said property refers to existential path quantification, said witness graph includes paths demonstrating only said property satisfaction, defining witnesses;~~

~~wherein said conclusive result is said property satisfaction when said property refers to said universal path quantification; and~~

~~wherein said conclusive result is said property violation when said property refers to said existential path quantification~~.

2. (currently amended): The method for verification as set forth in claim 1, wherein:

said generation of said testbench comprises:

determining embedded constraints for guiding vector generation based on said witness graph;

determining priorities for guiding said vector generation based on said witness graph;

generating a vector generator module including said embedded constraints and said priorities; and

generating a monitor module, said monitor module checking said conclusive result;

~~wherein, when said property refers to said universal path quantification, said vector generator module is generated so that said generated vectors are directed toward finding said counter-examples, and~~

~~wherein, when said property refers to said existential path quantification, said vector generator module is generated so that said generated vectors are directed toward finding said witnesses;~~ and

said simulation of said design, using said generated test bench, comprises:

generating said vectors with said vector generator module based on said embedded constraints, including generating random patterns and using said constraints as a filter to select desirable ones of said random patterns; and

checking said monitor module for property violation or satisfaction.

3. (original): The method of verification as set forth in claim 2, wherein:

said embedded constraints are derived from transition conditions in said witness graph; and

said priorities are associated with transitions in said witness graph.

4. (original): The method of verification as set forth in claim 3, wherein:

said priorities are generated from said witness graph based on one or more of:

distance to targets;

transition probabilities; and

simulation trace data.

5. (original): The method of verification as set forth in claim 1, wherein:

said generation of said witness graph comprises:

removing a portion from said design when an influence determination does not indicate that said portion of said design is in a cone of influence of said property;

modeling, as an initial abstract model, a controller state and variables in a datapath state directly involved in predicates of said correctness property;

performing deterministic analysis on said abstract model; and

pruning said abstract model to obtain said witness graph;

said influence determination indicates said portion of said design is in said cone of influence of said property when said portion of said design is one or more of:

a portion directly affecting said variables in said predicates of said property, and

a portion affecting branching which in turn affects predicates of said property;

said deterministic analysis determines which portion in said abstract model indicates paths relating to said conclusive result for said property:

said pruning comprises removing a portion in said abstract model indicated by said analysis not to relate to said conclusive result for said property.

6. (original): The method of verification as set forth in claim 5, wherein:

said pruning is followed by a step of refining said abstract model by adding variables from said datapath state to provide a refined abstract model;

said analysis, pruning, and refining steps are performed in an iterative process; and

said witness graph is said refined abstract model at the end of said iterative process.

7. (original): The method of verification as set forth in claim 5, wherein said property is represented using a computation tree logic (CTL) formula.

8. (original): The method of verification as set forth in claim 7, wherein said step of analysis is performed by:

determining CTL subformulas of said CTL formula;

with each of said CTL subformulas, associating a given set of abstract states corresponding to an over-approximate set of concrete states satisfying said CTL subformula, said given set of abstract states defining an upper set;

for ones of said CTL subformulas beginning with an E-type operator, performing standard model checking over said abstract model;

for ones of said CTL subformulas beginning with an A-type operator:

selecting an E-type operator corresponding to said A-type operator and guaranteed to result in an over-approximation, and

computing an other set of abstract states, corresponding to an intersection of said upper set with a set recursively computed for the negation of said A-type operator, said other set of abstract states defining a negative set;

checking an initial state of the design to determine a conclusive result, wherein:

when said initial state does not belong to said upper set, determining said property to be conclusively proved to be false;

when said property represented using said CTL formula starts with said A-type operator, and when said initial state belongs to said upper set, and when said initial state does not belong to said negative set, determining said property to conclusively proved to be true; and

determining said analysis to be inconclusive when said property is not conclusively to be proved to be one of true and false.

9. (original): The method of verification as set forth in claim 8, wherein said step of pruning comprises:

marking witness states; and then

pruning unmarked states by replacement with a sink state having every transition therefrom leading to said sink state, and all atomic propositions in said sink state being assumed false.

10. (original): The method of verification as set forth in claim 9, wherein said step of marking said witness states comprises:

computing a witness-top set of states consisting of the intersection of set of states reachable from initial state of said design and said upper set;

marking all of said witness-top set of states;

using a marking procedure, for each of said CTL subformulas of said CTL formula representing said property, with said witness-top set defining a care set, comprising the steps of:

associating with said CTL subformula, as a witness set thereof, a given set of states defined by the intersection of said upper set associated with said CTL subformula and said care set;

for ones of said CTL subformulas beginning with said EX operator:

marking additional states in the image of said witness set; and

recursively applying said marking procedure to the CTL subformulas thereof, beginning with said EX operator, with said additional states as said care set;

for ones of said CTL subformulas beginning with an A-type operator:

determining a neg-witness set as the intersection of said negative set associated with said A-type subformula and said care set; and

recursively applying said marking procedure on the negation of said A-type subformula, with said neg-witness set as said care set; and

for all other types of said CTL subformulas, applying said marking procedure recursively on the CTL subformulas thereof, with said witness set as said care set.

11. (original): The method of verification as set forth in claim 10, wherein said vector generator module is generated to include a search of said witness and neg-witness sets of states for a concrete witness, returning an indication of success when finding said concrete witness.

12. (original): The method of verification as set forth in claim 11, wherein said search is conducted suing a backtracking method comprising:

specifying a CTL formula;

specifying a given concrete state belonging to said witness set associated with said CTL formula;

starting from said concrete state;

determining an indication of success when there exists a concrete witness for said CTL formula, and failure otherwise; and

backtracking when said indication is failure, wherein:

when said CTL operator is not an A-type operator, search subproblems are conducted on the subformulas of said CTL subformula and each concrete state belonging to the associated witness sets;

when said CTL operator is an A-type operator and said state does not belong to said negative set, said indication is success;

when said CTL operator is an A-type operator and when said state belongs to said negative set, a search subproblem is set up with the negation of said CTL formula and said concrete state, wherein success of said negated subproblem indicates failure, and failure of said negated subproblem indicates success.

13. (original): The method of verification as set forth in claim 12, wherein:

said search subproblems on said CTL subformulas and said concrete states belonging to the associated witness sets are set up in a prioritized manner based on one or more of:

distance to targets,

transition probabilities, and

simulator trace data.

14-16. (canceled).

17. (currently amended): A ~~The~~ method of assessing simulation coverage of a given set of simulation vectors for a given design, comprising: ~~as set forth in claim 16,~~

providing a description of said design;

specifying correctness criteria for said design, wherein said correctness criteria are expressed as one or more correctness properties;

generating a witness graph for one or more of said correctness properties; and

determining coverage of said witness graph, using said given set of simulation vectors, by marking entities visited by said given set of simulation vectors in said witness graph, said entities being selected from the set consisting of states, transitions, and paths;

wherein: said generation of said witness graph comprises:

removing a portion from said design when an influence determination does not indicate that said portion of said design is in a cone of influence of said property;

modeling, as an initial abstract model, a controller state and variables in a datapath state directly involved in predicates of said correctness property;

performing deterministic analysis on said abstract model; and

pruning said abstract model to obtain said witness graph;

said influence determination indicates said portion of said design is in said cone of influence of said property when said portion of said design is one or more of:

a portion directly affecting said variables in said predicates of said property, and

a portion affecting branching which in turn affects predicates of said property;

said deterministic analysis determines which portion in said abstract model indicates paths relating to said conclusive result for said property; and

said pruning comprises removing a portion in said abstract model indicated by said analysis not to relate to said conclusive result for said property.

18. (canceled).

19. (new): The method for verification as set forth in claim 1, wherein said deterministic analysis comprises using symbolic model checking on the abstract model to identify the over-approximated states and transitions which capture all witnesses or counterexamples demonstrating the correctness properties.

20. (new): The method for verification as set forth claim 19, wherein said model checking further comprises:

determining sub-properties for said one or more correctness properties;

for each sub-property that can be represented with an operator applying to all paths in the abstract model, computing a negative set which is an intersection of states and transitions in the abstract model which satisfy a negation of the sub-property and states and transitions in the

abstract model which satisfy an over-approximation of the sub-property, such that the negative set can be used to identify states and transitions which capture witnesses or counterexamples for the sub-property.

21. (new): The method for verification as set forth in claim 20, wherein said one or more correctness properties are represented using a Computation Tree Logic (CTL) formula and wherein the sub-properties are CTL subformulas.

22. (new): The method for verification as set forth in claim 21, wherein the operator applying to all paths in the abstract model is an A-type operator.

23. (new): The method for verification as set forth in claim 21, wherein the operator applying to some paths in the abstract model is an E-type operator.

24. (new): The method for verification as set forth in claim 1, wherein said deterministic analysis comprises using constraint solving on the abstract model to identify the over-approximated states and transitions which capture all witnesses or counterexamples demonstrating said one or more correctness properties.

25. (new): The method for verification as set forth in claim 1, wherein the step of generating the witness graph further comprises the steps of:

refining the modified abstract model for the deterministic analysis by selectively bringing back datapath variables into the abstract model; and

using the deterministic analysis to prune the modified abstract model by removing states which cannot serve as either a witness or counter-example for said one or more correctness properties.

26. (new): The method for verification as set forth in claim 25, wherein the steps of refining and pruning are iterated until a limit on resources is reached.

27. (new): The method for verification as set forth in claim 1, wherein the testbench prioritizes the simulation search for said concrete witnesses or counterexamples based on said witness graph.

28. (new): A device-readable medium comprising program instructions for causing the device to perform verification for a design, comprising:

receiving an abstract model of said design, said abstract model abstracted from a design description of said design;

receiving one or more correctness properties representing correctness criteria specified for said design;

generating a witness graph for said one or more correctness properties based on a deterministic analysis of said abstract model where the deterministic analysis serves to modify

the abstract model by over-approximating states and transitions which capture all witnesses or counterexamples demonstrating said one or more correctness properties; and

where the deterministic analysis does not produce a conclusive result, automatically generating a testbench from said witness graph for performing simulation with said testbench, where the testbench generates simulation test vectors for the simulation that target the over-approximated states and transitions in the witness graph when searching for a concrete witness or counterexample with respect to said correctness criteria.

29. (new): The device-readable medium as set forth in claim 28, wherein said deterministic analysis comprises using symbolic model checking on the abstract model to identify the over-approximated states and transitions which capture all witnesses or counterexamples demonstrating the correctness properties.

30. (new): The device-readable medium as set forth in claim 29, wherein said model checking further comprises:

determining sub-properties for said one or more correctness properties;

for each sub-property that can be represented with an operator applying to all paths in the abstract model, computing a negative set which is an intersection of states and transitions in the abstract model which satisfy a negation of the sub-property and states and transitions in the abstract model which satisfy an over-approximation of the sub-property, such that the negative set can be used to identify states and transitions which capture witnesses or counterexamples for the sub-property.

31. (new): The device-readable medium as set forth in claim 30, wherein said one or more correctness properties are represented using a Computation Tree Logic (CTL) formula and wherein the sub-properties are CTL subformulas.

32. (new): The device-readable medium as set forth in claim 31, wherein the operator applying to all paths in the abstract model is an A-type operator.

33. (new): The device-readable medium as set forth in claim 31, wherein the operator applying to some paths in the abstract model is an E-type operator.

34. (new): The device-readable medium as set forth in claim 28, wherein said deterministic analysis comprises using constraint solving on the abstract model to identify the over-approximated states and transitions which capture all witnesses or counterexamples demonstrating said one or more correctness properties.

35. (new): The device-readable medium as set forth in claim 28, wherein the step of generating the witness graph further comprises the steps of:

refining the modified abstract model for the deterministic analysis by selectively bringing back datapath variables into the abstract model; and

using the deterministic analysis to prune the modified abstract model by removing states which cannot serve as either a witness or counter-example for the correctness properties.

36. (new): The device-readable medium as set forth in claim 35, wherein the steps of refining and pruning are iterated until a limit on resources is reached.

37. (new): The device-readable medium as set forth in claim 28, wherein the testbench prioritizes the simulation search for said concrete witnesses or counterexamples based on said witness graph.

38. (new): An apparatus for verification of a design comprising:

an input module for receiving an abstract model of said design, said abstract model abstracted from a design description of said design, and for receiving one or more correctness properties representing correctness criteria specified for said design;

a witness graph module for generating a witness graph for said one or more correctness properties based on a deterministic analysis of said abstract model where the deterministic analysis serves to modify the abstract model by over-approximating states and transitions which capture all witnesses or counterexamples demonstrating the correctness properties; and

a testbench module for generating a testbench automatically from said witness graph for performing simulation with said testbench, where the testbench generates simulation test vectors for the simulation that target the over-approximated states and transitions in the witness graph when searching for a concrete witness or counterexample with respect to said correctness criteria.

39. (new): The apparatus as set forth in claim 38, wherein said deterministic analysis comprises using symbolic model checking on the abstract model to identify the over-approximated states and transitions which capture all witnesses or counterexamples demonstrating the correctness properties.

40. (new): The apparatus as set forth in claim 39, wherein said model checking further comprises:

determining sub-properties for said correctness properties;

for each sub-property that can be represented with an operator applying to all paths in the abstract model, computing a negative set which is an intersection of states and transitions in the abstract model which satisfy a negation of the sub-property and states and transitions in the abstract model which satisfy an over-approximation of the sub-property, such that the negative set can be used to identify states and transitions which capture witnesses or counterexamples for the sub-property.

41. (new): The apparatus as set forth in claim 40, wherein said correctness properties are represented using a Computation Tree Logic (CTL) formula and wherein the sub-properties are CTL subformulas.

42. (new): The apparatus as set forth in claim 41, wherein the operator applying to all paths in the abstract model is an A-type operator.

43. (new): The apparatus as set forth in claim 41, wherein the operator applying to some paths in the abstract model is an E-type operator.

44. (new): The apparatus as set forth in claim 38, wherein said deterministic analysis comprises using constraint solving on the abstract model to identify the over-approximated states and transitions which capture all witnesses or counterexamples demonstrating the correctness properties.

45. (new): The apparatus as set forth in claim 38, wherein said generating the witness graph further comprises:

refining the modified abstract model for the deterministic analysis by selectively bringing back datapath variables into the abstract model; and

using the deterministic analysis to prune the modified abstract model by removing states which cannot serve as either a witness or counter-example for the correctness properties.

46. (new): The apparatus as set forth in claim 45, wherein the steps of refining and pruning are iterated until a limit on resources is reached.

47. (new): The apparatus as set forth in claim 38, wherein the testbench prioritizes the simulation search for said concrete witnesses or counterexamples based on said witness graph.